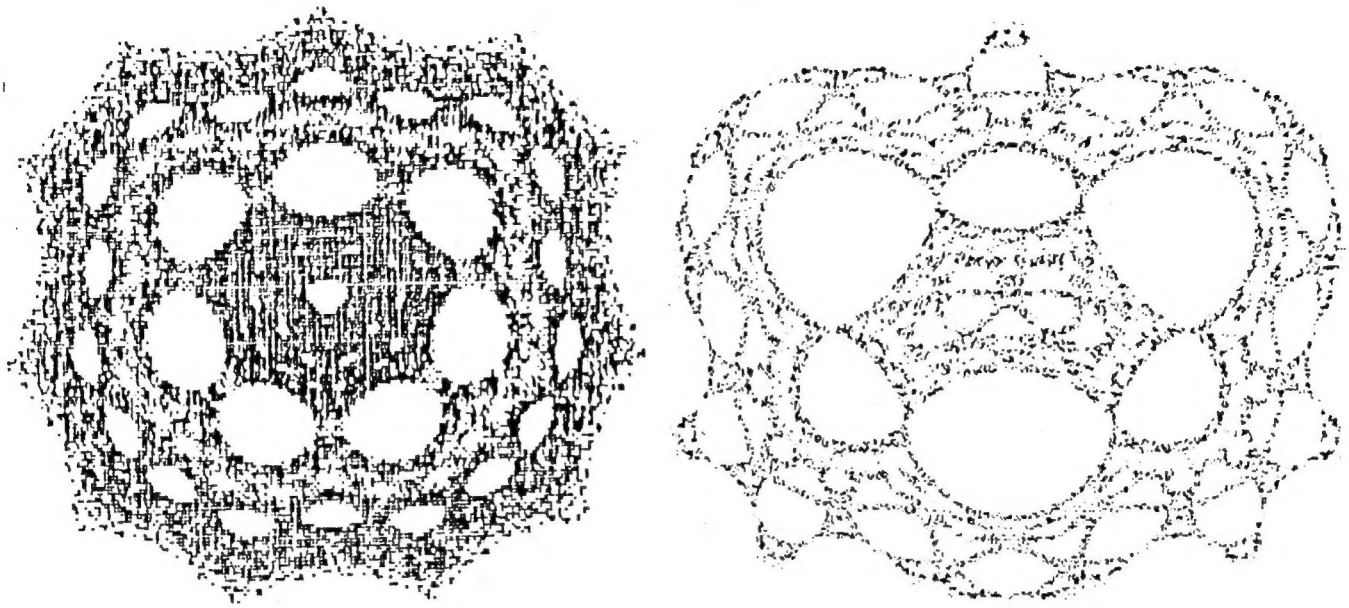

Above: Patterns generated by XYLEM - see page 3.

****

SPRITES - objects moving over a background

Reader N.V. French (Spalding, Lincs.) asked if it was possible to write a sprite routine in Beta Basic. He reported that he had almost managed to stop the background being rubbed out, using methods similar to Newsletter no. 9's "Putting Shapes on a Background", but was troubled by flashing. (Nasty, that!) Below is my attempt at a solution; it is a flash-free but fairly slow routine that moves an object over a background. I used a face floating randomly over part of the listing as an example - see the illustration. Read the issue 9 item again before attempting to understand the listing below, as I don't want to repeat my earlier explanation.

The disadvantage of the issue 9 method of placing an object on a background is that a "hole" has to be forced in the background to receive the object; momentarily, there is no object on the screen, and this creates flashing when you try to achieve movement. The method used by machine code games is to combine object and background data in registers, or a storage area, rather than on the screen. The combined data is then moved onto the screen without flashing. We can use the same method, although rather more slowly.

First, we need a method of keeping unaltered background in storage for later replacement when our sprite has moved on. This is more difficult than might at first appear, because if we simply use GET to store the screen image for the next intended position, we will store also the part of the sprite which currently overlaps the new position. I have implemented a sort of "conveyor belt" of background information to get round this; it starts with a copy of the background under the newly-placed sprite, and with every move a new record of a "clean" background is created from screen information, overlaid with part of the old record to wipe out the overlapping sprite. (Some games store an entire background screen - but this is memory-hungry.)

```
180 GET t$,x,y,3,3
190 ON ERROR 500
195 INK 7
200 PLOT bx(☺),t$
210 DO
```

Various combining operations require part of the screen (at bx,by) as a "scratch pad". The listing below allows you to see this area, plus the contents of the "conveyor belt"; to avoid this, remove the REM from line 195 so that the work area uses invisible ink, and delete line 230.

```
 10 LET x=11,y=163,r=9
 30 CIRCLE x,y,r
 40 FILL 11,167
 50 GET a$;0,175,3,3
 60 INVERSE 1
 70 CIRCLE x,y,r
 80 CIRCLE 8,166,1
 90 CIRCLE 13,166,1
100 PLOT 10,y
    PLOT 12,y
110 PLOT 7,159
    DRAW 8,0,1
120 GET b$;0,175,3,3
130 RANDOMIZE 50
    INVERSE 0
    CLS
140 LIST 180
150 LET bx=228,by=170
160 LET s=1,t=1
170 LET x=90,y=155
180 GET t$,x,y,3,3
190 ON ERROR 500
195 REM INK 7
200 PLOT bx,by,t$
210 DO
220   GET t$,bx,by,3,3
230   PLOT 0,175;t$
240   PLOT OVER 2;bx,by;a$
250   PLOT OVER 1;bx,by;b$
260   GET c$;bx,by,3,3
270   PLOT INK 0;x,y,c$
280   GET c$,x+s,y+t,3,3
290   PLOT bx,by,c$
300   PLOT bx-s,by-t,t$
310   LET x=x+s,y=y+t
320   IF RNDM(10)=1 THEN LET s=RNDM(4)-2,t=RNDM(4)-2
330 LOOP

500 INK 0
    GO TO 130
```

Notes: The DO-loop at line 210 is entered for the first time with "clean" background in the working area; this is stored in t$ using GET and held until it can be combined with screen background from the next position (in c$) at line 300. This gives a new "clean" background for the next passage through the loop. Line 320 sets up horizontal and vertical distances to move in variables S and T; replace it with keyboard control if you like. Line 150 use RANDOMIZE 50 because I found that this particular pseudo-random sequence will keep the sprite on-screen for some time; the ON ERROR at line 190 restarts the sequence when the sprite wanders too far.

****

XYLEM - a chaotic pattern generator

This contribution from Ettrick Thomson is based on a formula published (not for the first time) in Personal Computer World's Nov. 1987 issue. It has been used as a basis for pattern generation programs before. Ettrick's Beta Basic version has been more than doubled in speed by the use of a version of his machine code square root routine (see NL 4, p.13) which, incidentally, I am glad to be allowed to incorporate into SAM's ROM. Even so, the calculations required for each point mean a pattern takes 10-20 minutes to build up - or even longer for the higher-resolution (see NL 10) version I used for the front cover. You might want to take a coffee break!

```
 10 ON ERROR
      IF error=11 AND lino=110 THEN RETURN
      ELSE POP
        CONTINUE
 20 prologue
 30 INPUT "xos:";xos'"xrg:";xrg'"yos:";yos'"yrg:";yrg
 40 INPUT "p:";p'"q:";q'"r:";r
 50 PRINT p'q'r
 60 BORDER 4
 70 CLOCK "0"
    CLOCK 1
 80 LET x=0,y=0
 90 FOR i=1 TO 15000
100   LET z=y-SGN x*(p*x-q AND USR sqa),y=r-x,x=z
110   PLOT y+x,y-x
120 NEXT i
130 CLOCK 0
140 CLEAR -28

200 DEF PROC prologue
210   CLEAR 28
220   LET s=DPEEK(23730)+2
230   DO UNTIL PEEK (s-1)=0
240     LET s=s+3+DPEEK(s)
250   LOOP
260   FOR a=s TO s+27
270     READ b
        POKE a,b
      NEXT a
280   LET sqa=s
290   DATA 239,61,42,192,56,6,2,126,167,200,198,128,31,119,
      35,54,127,239,49,224,1,5,15,56,53,16,246,201
300 END PROC
```

Here are some parameters to try:

Pattern	xos	xrg	yos	yrg	p	q	r
1	63	135	51	102	17	11	5
2	125	257	90	180	31	13	2
3	64	142	57	114	3	5	7
4	100	256	88	176	5	11	17

Ettrick believes that the development of these patterns depends partly on the 'random noise' of rounding-off errors - the result varies noticeably according to the numerical precision of the language the pattern generator is written in.

****

SAM - THE NEW MICRO FROM MGT

Quite a few readers have noticed stories in the computer press revealing that I am writing the Basic for the new SAM micro, and asked for details. In fact, a number of readers suggested my name to MGT in the first place - thanks for the vote of confidence! I cannot reveal some things, partly because they are confidential, and partly because production machines aren't ready yet, and some details are not fixed. However, that still gives me plenty of scope.

The machine has been called a "Spectrum clone" which is a slur on a machine which completely outclasses the Spectrum. It is true that one screen mode is like the Spectrum's, and that the machine is flexible enough to run most 48K games. (You would be able to run even more software if you loaded a tape copy of the Spectrum's ROM into SAM's paged RAM.) It might be true to say that SAM is to the Spectrum what the Spectrum was to the ZX81. Certainly my idea of what a Basic should be has been influenced by ZX Basic - I happen to like syntax check on entry, string "slicers", automatic listings, etc. Clive Sinclair should have developed a machine like this, instead of the 128K Plus. Towards the end, he didn't seem to employ any imaginative engineers, though, and it has been left to a small, active outfit like MGT to fill the gap.

SAM has a Z80B "brain" chip, which runs 70% faster than the Z80A in the Spectrum. (If I write a particular Basic command no better than the Spectrum ROM version, I still get a 70% speed increase.) You get 256K or 512K of RAM, and further expansion is possible via the edge connector. The RAM is very cleverly paged to make life easy for machine-code types like me; Basic users will simply notice the nice features it allows me to provide, like BIG programs and arrays (100's of kilobytes), instantly switchable screens, fast operation, etc.

There are 4 screen modes. One major improvement in all modes is that the 16 colours we are used to on the Spectrum (8 bright, 8 non-bright) actually select an entry in a special palette memory; the contents of that memory then determine the colour you see on the screen. The 16 palette memories can be programmed with any of 64 different colours, mixed from 4 intensity levels for blue, red and green (4*4*4=64). With appropriate values, a game or screen picture will look the same as it did on the Spectrum, but often you would want to completely alter the colours. Even better, you can change the palette memory contents part-way down a screen, at any position, multiple times! This allows all 64 colours to be used at once, with some restrictions on location. Even the border colour can change part-way down a screen. Colours can also be made to flash - e.g COLOUR 1,30,45 would make colour 1 (normally blue on the Spectrum) flash between colour values 30 and 45. (I am not sure what actual names we will use for the colours yet - "red with a touch of blue and a dash of green?".) This can give pixel-level FLASH.

MODE 1: Just like the Spectrum's screen, using 6K for "pattern" data and 0.75K for attributes. Attribute clashing as usual!

MODE 2: This uses 6K for "pattern" data and 6K for attributes. Resolution is the Spectrum's 256*192 pixels, but any 8*1 pixel row in a character square can have its own Spectrum-style attributes.

MODE 3: Uses 24K. There is no separate block of attributes. Instead, each pixel has its own, as part of the main screen memory. Mode 3 has a resolution of 512*192 pixels, and allows 4 colours to be used at once from the palette of 16. (Which can be altered part-way down the screen.) You can have up to 85 6-pixel characters across the screen. I suppose this mode will be mainly used for word-processing and other detailed work. The use of any 4 colours is pretty good, and you can simulate more with stippled FILLs. However, my favourite is the next mode - mode 4!

MODE 4: Uses 24K. Resolution is 256*192, with 16 possible colours per pixel. It is very nice to get away from attribute problems! I recently ran a Mandelbrot set program to give complex, detailed colour patterns that would be quite impossible on the Spectrum.

Modes 2-4 all have nice easy memory arrangements for machine-code types - e.g. mode 4 uses 192 contiguous rows of 128 bytes, each half-byte coding for a pixel. To get a screen address from x and y coordinates takes 3 machine code instructions, vs. 20 or more on the Spectrum. This partly explains why e.g. DRAW is about 5 times faster on SAM than on the Spectrum. (Other reasons are the faster processor, and the fact that I thought very hard when writing the graphics commands!) Using half-bytes for each pixel is a good idea because it makes left and right ROLL and SCROLL by one pixel easily possible with the Z80's half-byte rotate instructions.

SAM's sound chip is better than the one provided on the 128K Spectrum, but I am not doing the programming on that part of the project, so I can't comment further on the sound facilities.

The keyboard will be "proper", with a set of function keys, and completely "soft" - that is, the keys will be programmable so that they can produce any character you want. This, combined with an extensive international character set, should make the machine particularly attractive in overseas markets.

SAM will be compatible with most Spectrum software, so it will start with a big software base, unlike most new machines. SAM Basic will do everything that Spectrum Basic can do, so Basic programs that don't try to be too clever with PEEKs, POKEs and USRs should run. A small and painless translation step is all that is required. As I write, I am not sure how many of Beta Basic's features will be packed into SAM's 32K ROM - all the major commands should be there, though, plus others to support SAM's hardware. What about compatibility with existing BB programs? Well, I cannot be completely tied to the current way of doing things, which are in some ways a "kludge" forced by the Spectrum's ROM. For example, using line zero to provide extra functions, and using UDGs as new keywords, will have to be ditched. But there are so many points of compatibility that you will normally be able to LOAD a BB program and then use some kind of SAM Basic translator program (full of ALTERs, I expect) to make the program runnable. The machine should be available from April 1989; the price will be about #140.00 for the basic 256K version. A compatible disc system will be available. MGT are allowing interested customers to reserve a place in the queue for machines at no charge - write to them if you are interested in doing this. Their address is:

MGT, LAKESIDE, PHOENIX WAY,
SWANSEA ENTERPRISE PARK, SWANSEA, SA7 9EH

****

IMPROVED 'ROSE' - faster patterns

This item causes me some embarassment; the item called "THE ROSE" in issue 11, p.2 works fine - but I had never got around to 'polishing' it before publication. Two readers, Gilbert Jackson and Ettrick Thomson, have contributed improvements which are major enough to justify another viewing. The use of BB's SINE and COSE, joining LET statements, and removing a completely redundant line 160 from the original give a large speed improvement. The variables N and D are now supplied by RNDM, in the range 1-179. (Values in the range 181-359 give essentially the same patterns.) RNDM showed up whole ranges of patterns that I had never seen before when choosing N and D myself. Line 40 incorporates an improved method of detecting repeating patterns in advance, to limit time-wasting overwriting.

```
 10 LET k=PI/180,xos=128,yos=88
 20 DO
 30   LET n=RNDM(178)+1,d=RNDM(178)+1
 40   IF MOD(n,2) AND MOD(d,8) THEN
        LET b=180
      ELSE LET b=0
 50   PRINT "N=";n'"D=";d
 60   PLOT 0,0
 70   LET a=0
 80   DO
 90     LET a=MOD(a+d,360),t=k*a,r=87*SINE(k*MOD(n*a,360))
100     DRAW TO r*SINE(t),r*COSE(t)
110   LOOP UNTIL a=b
120   PAUSE 0
130   CLS
140 LOOP
```

The above lines are self-contained and I urge you to try them out. I have also used this program in combination with the string-drawing procedures from issue 12, which I hope you have handy! It is possible to store about 40 different patterns in a string array, and then reproduce them at high speed - this makes a very impressive display. To do this, add or change the following lines in "The Rose" listing above:

```
  5 pokecd
    RANDOMIZE 99
    DIM p$(1)
 10 LET k=PI/180
 25   LET a$=""
 60   nplot 128,88
100     ndrawto r*SINE(t)+128,r*COSE(t)+88
120   LET a$=a$+CHR$ 254+CHR$ 254
      JOIN a$ TO p$
140 LOOP UNTIL MEM()<=3000
```

Line 5 pokes the DRAW-a-string code into place, selects a particular 'random' sequence (useful if you want to repeat something) and initialises an array which will have data for each pattern JOINed to it. The patterns are separated by CHR$ 254+CHR$ 254 - this won't occur in the pattern data, so we will be able to recognize these separators later using INSTRING. Once p$ is big enough to fill a lot of memory, we stop adding patterns to it. You can add a line to save the array at this point, e.g.:

```
150 SAVE 1;"patdat" DATA p$()
```

Add the following lines to find the stored patterns and re-draw them at high speed:

```
200 LET p=2,s=p
210 DO
220   CLS
230   LET p=INSTRING(p,p$,CHR$ 254+CHR$ 254)
240   LET a$=p$(s TO p-1)
250   sdraw a$
260   PAUSE 0
270   LET p=p+2,s=p
280 LOOP UNTIL p=LEN p$+1
```

Finally, MERGE in procedures POKECD, SDRAW, NPLOT and NDRAWTO (from issue 12). You can have a program that includes only lines 200-280, POKECD and SDRAW, and a line to load the array p$() and call POKCD, to give a very nice demo.

****

PROC CARTESIAN - converting polar to X,Y coordinates

Ettrick Thomson (Aldeburgh, Suffolk) commented that my discussion of passing parameters by reference (NL 11,p.4) didn't mention an advantage of REF; it allows you to handle a function that can't use DEF FN since it can't be specified by a since assignment statement. It also allows you to handle more than one function. Ettrick's example procedure takes as its first two parameters the names of the variables to put the output values in, followed by a radial distance and a bearing. (I took the liberty of making a slight modification so that the bearing can be in degrees, not radians - I still think this way, as I suspect most of us do.) Range and bearing coordinate systems are called "polar". You are probably familiar with at least some of their uses; e.g. "Torpedos bearing 150, 200 yards and closing, Captain!". Computers use the more familiar X,Y system - rectangular, is that the term? So we need to do a conversion in order to plot a pixel. Note that the X and Y of line 30 can be altered to something else - we are just telling the procedure what variable to put the answer in! (Line 40 would have to change too, of course.)

```
 10 LET xos=128,yos=88
 20 FOR a=0 TO 359 STEP 10
 30   cartesian x,y,50,a
 40   PLOT x,y
 50 NEXT a

100 DEF PROC cartesian REF x, REF y,r,theta
      LET x=r*SINE(theta*PI/180),y=r*COSE(theta*PI/180)
    END PROC
```

****

FACTORIALS revisited.

Ettrick also came up with the recursive function method of calculating factorials that I was trying to think of in NL 11, p. 5. He had it published in Computer Bulletin in June 1985!

```
DEF FN h(n)=VAL(("1" AND n=0)+("n*FN h(n-1)" AND n>0))
```

He says: "Explaining how FN h(n) is calculated is a nice problem in analysing ROM behaviour!". Very true - 'advanced topic', I'd call that! Note: the 'FN' must be a keyword.

****

NARROWER CHARACTER SET

This contribution is from Neil Exley of Carnforth, Lancs. He writes:

The program below redefines the Spectrum character set, to make the characters appear slightly smaller than usual. It is particularly useful with CSIZE 6,8, instead of using the normal size character set and OVER 1. The program takes about half a minute to complete, so don't worry if the screen goes blank for a while. Line 60 makes the new character set appear on a 40 column screen, as what you would find on other computers.

```
10 CLEAR 384
   CLEAR 384
   LET p=DPEEK(23730)+2
   DO UNTIL PEEK (p-1)=0
     LET p=p+3+DPEEK(p)
   LOOP
   POKE p,MEMORY$()(15616 TO 16383)
   DPOKE 23606,p-256
20 FOR a=1 TO 16
     READ x,y
     FOR b=p+x to p+y
       LET b$=BIN$(PEEK b),a$=b$( TO 3)+b$(5 TO )+"0"
30     IF b$(4)="1" THEN LET a$(4)="1"
40     POKE b,VAL (CHR$ 196+a$)
     NEXT b
   NEXT a
50 DATA 16,31,40,47,128,159,168,207,232,239,248,327,336,41
   5,424,455,464,479,512,559,568,583,624,671,680,687,712,7
   19,728,751,760,767
60 WINDOW 1,0,175,241,176
   WINDOW 1
   CSIZE 6,8
   CLS
   LIST
```

****

STRING AND NUMBER INPUT PROCEDURES

John Watkins (London) writes:

The contributions are an improvement over the string/number PROCedures in issue 3/4. Mine seem to operate much quicker (in the case of the string input) and in the case of the number input offers better error checking, especially with respect to exponents and negative numbers.

```
6000 DEF PROC String REF b$,length,a$
6010   DEFAULT a$=""
       LET b$=a$
6020   PRINT b$
6030   DO
         IF LEN b$<length THEN PRINT "_";CHR$ 8;
6040     GET a$
       EXIT IF a$=CHR$ 13
6050     IF CODE a$>31 AND CODE a$<128 AND LEN b$<length THEN
           PRINT a$;
           LET b$=b$+a$
6060     IF CODE a$=12 AND LEN b$>0 THEN
           LET b$=b$( TO LEN b$-1)
           PRINT CHR$ 8;"  ";CHR$ 8;CHR$ 8;
6070   LOOP
6080   IF LEN b$<length THEN PRINT " ";
6090 END PROC
```

In PROC number most is self-explanatory, finnum being the exit number, length the number of characters long 'b$', a temporary string is. Min and Max are obvious, and the other three are logical variables to denote the range of the number. Unfortunately any error trapping relating to line 6250 produced some odd results and infinite (Basic) loops.

```
   10 PRINT AT 0,0;
      Number num,10,0,1e6,1,0,1,""
 6100 DEF PROC Number REF finnum,length,min,max,dec,neg,expo,b$
 6110   LOCAL a$
 6120   DEFAULT expo=0,dec=0,b$="",neg=0
 6130   PRINT b$
 6140   DO
 6150     IF LEN b$<length THEN PRINT "_";CHR$ 8;
 6160     GET a$
        EXIT IF a$=CHR$ 13
 6170     IF CODE a$>47 AND CODE a$<58 AND LEN b$<length THEN
            PRINT a$;
            LET b$=b$+a$
 6180     IF a$=CHR$ 12 AND LEN b$>0 THEN
            PRINT CHR$ 8;"  ";CHR$ 8;CHR$ 8;
            LET b$=b$( TO LEN b$-1)
 6190     IF neg AND a$="-" AND LEN b$=0 THEN LET b$=a$
            PRINT a$;
 6200     IF dec AND a$="." AND INSTRING(1,b$,".")=0 THEN
            LET b$=b$+a$
            PRINT a$;
 6210     IF LEN b$=0 THEN GO TO 6240
 6220     IF expo AND INSTRING(1,b$,"E")=0 AND (a$="E" OR a$="
          e") THEN
            LET b$=b$+"E"
            PRINT "E";
 6230     IF expo AND a$="-" AND b$(LEN b$)="E" THEN
            LET b$=b$+a$
            PRINT a$;
 6240   LOOP
 6250   LET finnum=VAL b$
 6260   IF finnum<min OR finnum>max THEN BEEP .1,.1
          GO TO 6140
 6270   IF LEN b$<length THEN PRINT " ";
 6280 END PROC
```

****

BETA BASIC AND THE PLUS 2A

The PLUS 2A? What's that?! Well, it seems Amstrad have run out of Plus 2 circuit boards - but they had planned for this when they designed the Plus 3 circuit board. This board can run with a built-in tape recorder instead of a disc drive, so they have stuck Plus 3 boards into Plus 2 cases to produce the Plus 2A. Unfortunately, the new board uses 64K of ROM that is quite different from the 32K ROMs in the Plus 128K or Plus 2, and Beta Basic 4.0 will not run. I will not have the time to solve this, unfortunately. If I'd accidentally bought a Plus 2A, I would feel rather annoyed, because there is no warning on the machine that it is NOT a Plus 2, and some peripherals and programs will not work. The new machines can be spotted by their different case colour - black, rather than grey.

A recent BB 4.0 buyer and subscriber (Stan Flynn, Stafford) who turned out to have a Plus 2A found a solution to the incompatibility of BB 4.0. He was helped out by a friendly hardware person, and as far as I can guess from the description, 2 ROM chips were pulled from the Plus 2A board and replaced by a single Plus 2 ROM. The machine then acted like a Plus 2 and

BB 4.0. I don't know if the chips were socketed, rather than soldered in place, though I'd guess so; I don't know which vacated socket the Plus 2 ROM was put into, either. But there are probably suppliers of Plus 2 or Plus 128K ROMs around, so this is one option to consider if you want to run BB 4.0 and have problems getting a Plus 2 or Plus 128K. Note: It probably wouldn't matter if you tried the chip in the wrong socket - the machine would just fail to work - but you MUST line up the notch or dot depression at the end of the chip the same way as the original chip was lined up! (I once made this mistake with an EPROM and it lit up like a little light-bulb inside its glass window... amazingly, it still worked after this!)

****

PROC EXAM - examine memory

This contribution is from Terry Holland of Whitley Bay, Tyne & Wear. He says:

"EXAM 'peeks' into a specified number of addresses and displays the contents thereof. Parameter a is the starting address (defaulted at 23755), and c the required number of addresses (defaulted at 400). I hasten to add that this is not all my own work. I adapted this from a routine contributed by a reader to one of the popular Spectrum magazines."

```
 10 DEF PROC EXAM a,c
 20   CLS
 30   DEFAULT a=23755,c=400
 40   LET n=1
 50   DEF FN b(i)=INT (i/2)=i/2
 60   FOR i=0 TO c
 70     POKE 23692,0
 80     LET b=FN b(n+i)
 90     BRIGHT b
100     PRINT n+i;TAB 4;a+i;TAB 10;PEEK (a+i);TAB 15;CHR$
        (PEEK (a+i)) AND PEEK (a+i)>31
110   NEXT i
120   BRIGHT 0
130   POKE 23692,2
140 END PROC
```

Editor's notes: The function B(i) is used to find if the address is odd or even to give a display with bands of BRIGHT printing. Care is taken at the end of line 100 to avoid printing control codes (values less than 32). I thought that the suppression of the 'Scroll?' prompt using POKE 23692 was a disadvantage here - but see what you think. Looking at a Basic program this way will show you lots of fairly confusing stuff, like invisible forms of numbers (preceded by CHR$ 14) and two-byte line numbers (usually preceded by the CHR$ 13 that ends the previous line) followed by line length data. You will be looking near the start of the program using Terry's default values, which is BB's line 0. This is fairly boring - try about 24404 to see the first line after.

****

THE MEMORY REQUIREMENT OF ARRAYS

A recent letter leads me to think that not everyone knows how to calculate the memory requirements of an array. String arrays are fairly simple; e.g. DIM a$(30,10) would use 30*10=300 bytes, plus a few for information used by the system. However, creation of a similar array of numbers e.g. DIM a(30,10) would use over 1500 bytes, because each number takes 5 memory locations. You can see why it might be worth considering using CHR$, or CHAR$, to code numbers as 1 or 2 character strings!

****

MUSICAL SIMON - guess a note

Terry Holland (Whitley Bay, Tyne & Wear) says:

"SIMON is a sort of game. The computer sounds a note (a 48K beep) and the user must identify and sound the same note as quickly as possible. Beta Basic's 'stopwatch' facility is used here. The user's time is indicated including the time to beat."

```
 10 DEF PROC SIMON
      POKE 23609,0
      LET N$="00:00:99"
      LET Y=99
 20   DO
        CLOCK 0
        CLS
 30     LET NUMBER=RNDM(7)+1
 40     PRINT "IDENTIFY THE NOTE "
        PRINT "------------------"
        PRINT "THAT I WILL SHORTLY"
 50     PRINT "------------------"
        PRINT "SOUND BY PRESSING"
        PRINT "------------------"
        PRINT "KEYS 1 TO 8"
 60     PRINT "------------"
 70     PRINT
 80     LET Z$=N$(5)+N$(7 TO 8)
        LET X=VAL Z$
        IF X<Y THEN LET Y=X
 90     PRINT "THE TIME TO BEAT IS ";Y
        PRINT
100     PAUSE 100
110     ON NUMBER
          BEEP .3,0
          BEEP .3,2
          BEEP .3,4
          BEEP .3,5
          BEEP .3,7
          BEEP .3,9
          BEEP .3,11
          BEEP .3,12
120     PRINT "GUESS NOW "
        POKE 56866,5
        POKE 56870,58
        CLOCK 1
        CLOCK "0"
130     DO
          GET GUESS
          ON GUESS
            BEEP .3,0
            BEEP .3,2
            BEEP .3,4
            BEEP .3,5
            BEEP .3,7
            BEEP .3,9
            BEEP .3,11
            BEEP .3,12
140       IF GUESS=NUMBER THEN LET N$=TIME$()
            CLOCK 0
            PRINT "WELL DONE YOU HAVE GUESSED"
            PRINT "CORRECTLY IN A TIME OF ";N$(5);N$(7 TO 8)
          ELSE PRINT "WRONG TRY AGAIN"
```

```
150      LOOP UNTIL GUESS=NUMBER
160      INPUT "DO YOU WISH ANOTHER TRY - ENTER ANY KEY FOR Y
         ES OR N FOR NO ";A$
170    LOOP UNTIL A$="N" OR A$="n"
180    POKE 23609,32
       POKE 56866,50
       POKE 56870,54
       CLOCK 0
190 END PROC
```

Editors note: The POKEs to BB's CLOCK to make it act as a 'stopwatch' are given on page 18 of the BB 3.0 manual. Address 56866 can be poked with values as small as 1 for a faster clock. (SIMON doesn't worry about what units it is timing in, anyway.)

****

GENERATING DEF FNs

This contribution is from the prolific W. Ettrick Thomson (Aldeburgh, Suffolk). The procedure uses KEYIN to generate a program line containing a DEF FN. This brings up a general point; if you make a syntax error using KEYIN, you are not given the chance to edit the line but are presented with a 'Nonsense in Basic' report. The procedure shows a way out, using ON ERROR. The methods used here are applicable to writing something like a full-screen editor using SCREEN$ and KEYIN. (I know - I've got a sort-of-working example!)

Why do you need a procedure to generate DEF FNs? After all, you can just type them in. Well, it might be convenient to allow a user to create a function - for example, in a graph-plotting program - without telling him he has to break into the program and type it in somewhere! The procedure allows the DEF FN creation from within the program. One warning - the version shown here works fine if you just type 'function', but if it is included in a program, the line number that the DEF FN is placed at should higher than 20. The insertion of a new line near the top of the listing alters return addresses used by procedures, so the DEF FN must be KEYed in after all procedures that are being used at the time (not including PROC function).

```
200 DEF PROC function
210   ON ERROR
        IF error=12 THEN
          PRINT '"Syntax error in"'" DEF FN f(x)=";g$'
          "Please re-enter"
          LET f=1
          RETURN
        ELSE POP
          CONTINUE
220 PRINT "--Solve the equation y=f(x)=0--"'
    PRINT "f(x) must be expressible"
    PRINT "as a BASIC FN."
    PRINT "Please complete the DEF FN."
230   LET f=0
      INPUT "DEF FN f(x)="; LINE g$
240   KEYIN "20 def fn g(x)="+g$
250   IF f THEN GO TO 230
260   ON ERROR 0
270 END PROC
```

****

READERS' LETTERS AND SUCH...

Welcome to issue 13 of the Newsletter! I got a LOT of very gratifying comments from readers as they renewed their subscriptions - thank you all very much. These letters have convinced me that basically the Newsletter content is about right for those who renewed - unfortunately, quite a few subscribers have dropped out, but it is too late to try to please them. (A few wrote, saying Bye-bye, I've bought an ST/Amiga/PC am emigrating/etc. Quite a few pleas to implement BB on other machines have arrived, too.) Actual critical comments were very hard to find, except on one point which I will come to later. One reader suggested that my recent illustrations were a waste of space, and one said stuff on BB 4.0 was wasted on him because he only had BB 3.0. One person wanted more long contributions, in sections, and another said he wanted more machine code subroutines. However, the main criticism is exemplified by the following letter:

DEAR ANDY!

I think the BB Newsletter is very good, but is it possible and okay with you if the non-hesitant pay for one year the next time? The problem is that a check for 3.50 pounds costs almost 3.50 pounds to purchase. Even if I have ancestors that are not Scottish like yours (see BB no. 3 last page) they come from Smaland a province in southern Sweden and they are famous for the same thing as the Scotsmen namely to be very economical.

Hans Muhrbeck, Taby, Sweden

I got LOTS of letters in this vein - sorry for the cost and inconvenience! Some subscribers side-stepped the problem by simply sending more money, for 6, 9 or 12 issues, which I hadn't expected. Why the change to 3 issues? Well, I hardly dare write this... I had seriously considered abandoning the Newsletter. Contributions had been a little thin during the previous issues, and I had had to write quite a few things myself. I enjoy this, but it takes lots of time. I made the mistake of estimating the time taken per Newsletter, vs. the financial returns. This turned out to be pretty low - better than being unemployed, but not as lucrative as, say, being a cleaner. Actually, the best returns from editing the Newsletter are not financial, but the satisfaction of communicating with a lot of interesting and interested people, so you might think I am being a bit money-grabbing and the Newsletter should just be run as a hobby. The problem here is that since I am self-employed, the Newsletter competes fairly directly with normal income-generating activities, which are fairly important. (If I go on holiday, or get sick, I don't get paid.) I got married since the last issue, too, and this tends to prompt a certain amount of thought on financial matters.

My thinking was: "What if I am committed to another 6 issues, contributions dry up, and lots of subscribers drop out? I would have months of work for virtually no return." I half-decided to terminate the Newsletter, but in the end, I had been getting lots of appreciative correspondence, and I found I just couldn't do it. I compromised by reducing the number of issues I was comitted to. (Readers who sent money for lots of issues have been carefully noted so they won't lose out whenever publication ceases - which it will do eventually, of course. Nothing lasts forever!) In the event, I got a good response to my appeal for

contributions, and I may include some of them without typing them in myself, in order to save time. I hope you will excuse any reduction in legibility. (Most of this issue was put together before many of these contributions were received, but they will appear in later issues.) On the other hand my printing costs have just jumped by almost half, so I was pleased that I'd increased the cost of each issue a bit!

Dear Dr. Wright,

I have found a problem with SPLIT using BB 3.0+D, the example on page 63 of the (excellent) manual doesn't give the expected results - see attached printout. There is no problem unless the line being edited contains numbers to the left of the <>... I hope there is a simple solution.

John Fellowes, Norwich

The problem that you and fellow-reader David Oliver pointed out only exists in the Disciple/Plus D version of Beta Basic. I found that one of my modifications to cope with the Disciple's habit of stripping the invisible forms of numbers from Beta Basic statements caused the problem. Fortunately, a simple POKE corrects things, although the action of SPLIT is slightly altered from the description given in the manual. After POKE 58539,89 (for either BB 3.0+D or 4.0+D) SPLIT should work without the problems you mentioned, but when the first part of the split line has been entered, your cursor will be left at the start of a line which has NO line number, rather than the same number that the first part has. In some ways this may be better than the original system, so I hope you can live with it! Users of non-Disciple versions of BB can make this POKE,too.

Dear Dr. Wright,

If bbc2 CODE is 6144 bytes long then how when loaded to the RAM disc does it occupy only 1K of space? The only explanation I find is that the extra 5K go into some space in RAM disc not available from Basic.

Ricardo Cohn, Montevideo, Uruguay

You've nearly got it. There is 80K of paged RAM available on a 128K Spectrum. Only 74K of this is used for the RAM disc; the rest is filled by the editor's work area and a disorganised clutter of variables, routines and free memory. Five K from the file "bbc2" CODE overlay this area, so that only 1K has to be taken from the RAM disc area.

Dear Dr. Wright,

I enclose five pounds... with the enormous extra 1.50 I hope to bribe you into letting loose those Opus routines that you have been talking about earlier. Should you have hints of magazines or user-clubs, I would be very grateful.

Finn Hansen, Esbjerg, Denmark

I am very sorry, but I can't find the time to search through my pile of discs and tapes to print any Opus Discovery programs I might have. The most vigorous user-group I know of is FORMAT, which started as a Disciple/Plus D magazine. It is moving to attract non-Disciple readers in an effort to become the last general-readership Spectrum magazine not totally dedicated to games. It is very professionally produced (much more so than the BB Newsletter) by Bob Brenchley, and runs to about 32 pages every month. It carries reviews, news, SAM info, an adventure corner, articles on disc drives, Basic and machine code - I even contribute the odd item on Beta Basic! (Which will probably have appeared in some form in the BB Newsletter already.) It is well worth a look. Write or phone for information to:

FORMAT, 34 Bourton Road, Gloucester GL4 0LE. Tel. 0452-412572

Dear Dr. Wright,

I have had much entertainment from your rotating cube and superfast plot/draw programs (NL no. 12). A minor question is "Why FOR n=1 TO 0 at line 550 page 4?". I am intrigued by the last 5 bytes of line 550 page 9 (LD HL 2758H, EXX, RET). They can be replaced by: EXX, POP HL, EXX, RET if the code is started with EXX, PUSH HL, EXX so the purpose appears to be to restore the HL' pair after the use of EXX within the program (e.g in the LINE-DRAW routine in ROM). What puzzles me is the absence of an EXX before the LD HL, and, also, the fact that one can be inserted there without ill effect. That one can play ducks and drakes like that with the alternate registers like that strikes me as very odd.

G. Jackson, Cardiff

The dummy FOR-NEXT line is because at line 560 variable b is assigned the address of the array f$; when line 580 is reached, the creation of n by the "real" FOR statement could move the array! So I made sure that n already existed, as a FOR variable, before the value of b is set. The alternate register set query came from several readers, so my machine code is being taken to bits by quite a few people, obviously. (How gratifying!) I used the method I did because it takes 4 bytes, which is less than any alternative I can think of. Many assembly-language programmers never use the alternative set of registers (called HL', DE' and BC') at all - they are often not particularly useful. However, they are used by DRAW, and this can cause a crash when DRAW is used via USR. The ROM expects the HL register that is "switched out" to contain 2758H on return from a USR (this is the address of an "exit floating point calculator" code which has to be executed after USR). There is no way to tell which set of registers is which, other than by guessing on the basis of their contents, so my method is as good as saving the "real" HL' pair and restoring it later, because in a sense there isn't actually a "real" HL' pair - just two sets of registers that work identically. As long as the contents are sensible, all is well. (I will make sure SAM saves any registers that matter!)

Dear Andy,

In the last para. of p.16 of NL 10, you mention that 'INPUT ""' can be used to clear the lower screen and reset the 'Scroll' counter. 'INPUT ;' does the same thing, but is one byte shorter.

W. Ettrick Thomson, Aldeburgh, Suffolk

Dear Dr. Wright,

I wonder if you can offer any advice? I retained my 48K when I obtained my +2. The 48K is linked to a Microdrive and teletext adapter and the +2 to an Opus drive. And of course they both have an RS232 port. So they can pass information. Or that's what I thought. I have had no success yet. Unfortunately the +2 manual whilst admirable in many respects is very vague on this point and when I wrote to Amstrad for advice they did reply but added nothing to clarify the techniques. Is it in fact possible or is the +2 port nor 'real' in the sense of being usable in this way? I would like to pass screens, code and programs but have only had failure to date. I wonder if you or any of your readers has any suggestions to make?

Richard Lown, Wivenhoe, Essex

I use my Plus 128K all the time to send data back and forth to an Amstrad CPC (fitted with an external RS232 interface) and a PC. I often want to send all characters and control codes without alteration, so I usually start by POKE 23349,39: POKE 23350,1 to allow this. (See BB 4.0 manual, page 17 - Crash's Tech Niche *gave a #30.00 prize to someone for this tip - the year AFTER BB 4.0 was launched!) You would want to do this to send most kinds of data, except listings (as opposed to runnable programs). Set the baud rate for the RS232 using FORMAT "p";(baud rate). Most computers seem to use 9600 by default. You can open a stream to the "p" channel, and PRINT to that stream, or just LPRINT the characters you want. If the other computer isn't ready to receive, the sending computer should wait; if it doesn't, there is probably something wrong with the cable. To receive data, open a stream to the "p" channel, and INPUT or INKEY$ from that stream. That's fine with data or machine code files - you can use PRINT and INPUT or INKEY$ fairly easily, but what about programs? Interface 1 allowed SAVE *"b" and LOAD *"b". (I got out the manual to check, since my Interface 1 is not set up permanently.) I thought, "Pity they didn't implement that on the 128K Spectrums" and just on the off-chance I tried it. I got very excited when it worked! Then I pulled my Opus of the back and it stopped working... Opus Discovery allows SAVE *"b" to send programs via the PARALLEL printer port! However, it also tried to send and receive something via RS232 when I tried SAVE*"p" and LOAD*"p", so you might be in luck! The CPC was getting something, but naturally couldn't make sense of it. Try your Interface 1 instead. Remember to make the POKEs at the 128K end of the link.*

Dear Dr. Wright,

What is the news about BB on the Plus 3 and other computers?

Bernard Wild, Colchester, Essex

(Bernard's letter is reconstructed, since I cannot put my hand on the original!) I will probably never finish the Plus 3 version of BB - I am too busy writing the Basic for SAM - sorry! I can recommend the Plus D disc Interface if you want to add a disc drive - there IS a Beta Basic version for that. Or wait for SAM. I thought of writing BB for the CPC and PCW machines. I got as far as getting Spectrum Basic to work on the CPC and selling some tapes, but didn't really market the product - too much work and expense for the likely return. (However, I do find this program very useful for playing with altered versions of the Spectrum ROM, since the "ROM" is loaded into editable RAM.)